75

Popular Computing

The world's only magazine devoted to the art of computing.

June 1979 Volume 7 Number 6

50	1.1258999068842624	15
100	1.267650600228229	30
150	1.427247692705960	45
200	1.606938044258990	60
250	1.809251394333066	75
300	2.037035976334486	90
350	2.293498615990072	105
420	2.707685248164858	126
450	2.907354897182428	135
500	3.273390607896142	150
600	4.149515568880993	180
720	5.515652263101987	216
810	6.828046779268971	243
900	8.452712498170644	270
1000	1.071508607186267	301
1500	3.507466211043404	451
1710	5.771551634922064	514
2000	1.148130695274255	602
2500	3.758280234548012	752
3000	1.230231922161117	903
3600	5.104866514341946	1083

A table of powers of 2

Each power is given by the first 16 significant digits and the power of 10 to locate the true decimal point.

For example, the 1000th power of 2 is an integer of 302 digits.

4000	1.318204093430943	1204
4200	2.118272307832115	1264
4500	4.314996898727097	1354
4800	8.789803920478832	1444
5000	1.412467032139426	1505
5500	4.623556316968135	1655
9000	1.861919823602447	2709
10000	1.995063116880758	3010

1.41421356237309504880168872420969807856967187537694807317667973799073247846210703885038753432764157273501384623091229702492483605585073721264412149709993583141322266592750559275579995050115278206057147010955997160597027453459686201472851741864088919860955232923048430871432145083976260362799525140798968725339654633180882964062061525835239505474575028775996172983557522033753185701135437460340849884716038689997069900481503054402779031645424782306849293691862158057847311159666871301301561856898723723528850926486124949771542183342042856860601468247207714358548741556570696776537202264854470158588016207584749226572260020855844665214583988939443709265918003113882464681570826301005948587040031864803421948972782906410450726368813137398552561173220402450912277002269411275736272804957381089675040183698683684507257993647290607629969413804756548237289971803268024744206292691248590521810044598421505911202494413417285314781058036033710773091828693147101711116839165817268894197587165821521282295184884 7...

Shown above are the first 1000 significant digits of the square root of 2.

The square of this 1000-digit number is a 2000-digit number having the form 1.(998 nines)...

If the 1000th digit is changed from 7 to 8, then the square will have the form 2.(998 zeros)..., thus proving the result.

The square root of 2 is known to one million decimal places. The only other irrational number known to that level of precision is pi.

★★★★★★★★★★★★★★★★★★★★★★★★★★

Publisher: **Audrey Gruenberger**
Editor: **Fred Gruenberger**
Associate Editors: **David Babcock**
Irwin Greenwald
Patrick Hall
Contributing Editors: **Richard Andree**
William C. McGee
Thomas R. Parkin
Edward Ryan
Art Director: **John G. Scott**
Business Manager: **Ben Moore**

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $20.50 per year, or $17.50 if remittance accompanies the order. For Canada and Mexico, add $1.50 per year. For all other countries, add $3.50 per year. Back issues $2.50 each. Copyright 1979 by POPULAR COMPUTING.

Apple II

The Apple II personal computer has been marketed for nearly two years as one of the top-of-the-line machines. The processor and all needed circuitry is contained in a plastic case (15 x 18 x 5 inches) including a 52-key typewriter-like keyboard. To make a complete system, the user adds a cassette player and a TV monitor. The Apple features color control of the TV screen, and color demonstration routines are included with each machine that are spectacular. RAM storage is available in 16K-byte increments from 16K to 48K (it is possible to buy an Apple with only 4K of RAM, but few people would). Integer BASIC is standard in ROM. Floating point BASIC ("Applesoft") can be obtained in ROM, or is furnished on cassette. The company advertises "6502 assembly language" as standard. A monitor that is quite powerful is built into a ROM. Facilities for expansion (to dual floppies, for example) are available.

Much of the software that is supplied with the machine comes with little or no documentation. The machine reference manual (the red manual) is a monumental glitch from cover to cover, featuring up to half a dozen typos per page, and evidently written by a 12-year-old. It has no index.

The phrase "symbolic assembler" is so common that one tends to forget that there can be an assembler that is not symbolic. But that is what Apple offers in its firmware; a bare-bones minimal assembly program and a disassembler. It's clever, neat, and inexpensive, but it's miles from what one expects for an assembler. If you want to use the speed and power of the 6502 processor, be prepared to work at the bit level.

It is fairly easy to get to the 6502 level in the Apple, and the processor operates in the machine at the rate of 250,000 executed instructions per second.

The two BASIC interpreters that come with the Apple are magnificently incompatible, almost to the point where it seems to have been done deliberately. For example, one of them has the MOD function, while the other one doesn't. Neither of them has any kind of usable STOP command. Even the horizontal and vertical spacings on the screen are radically different. The point is that Apple had the opportunity to do these things right; they bought the floating version from Microsoft and could have insisted that it be made compatible wherever possible.

Now, one must agree with Emerson's belief that "A foolish consistency is the hobgoblin of little minds." But Emerson wrote no essays on intelligent consistency, nor on a passion for inconsistency for its own sake. Consider, for example, the RND function. In integer BASIC, this function produces integers in the range from 0 to 32767; in the floating point system, it produces fractions, X, in the range $0 \leq X < 1$. So far so good--that's what you'd want it to do. But the integer RND is not a random number generator at all; it scrambles the numbers from 0 to 32767 (that is, each of those numbers appears once and only once in each generation cycle). Thus, it generates random permutations, and with an extremely short cycle.

The floating point BASIC is full of odd little quirks. For example, the (perfectly legal) statement:

```
100 IF X< A  THEN 200
```

lists (and refuses to execute) as:

```
100 IF X< AT  HEN200
```

(but any letter but A in the statement works properly).

The complete assembly language listing of the monitor is supplied, but it is poorly documented. To find out such fundamental things as how to implement carriage return and line feed (those are quaint terms to apply to a CRT), one asks a friend. It's in that monitor, but you'll never find it out by yourself.

To be sure, the Apple users form a close-knit group (this is true of users of TRS-80s, Porsche's, and various brands of surfboards, too), but it is disconcerting to hear a casual remark like "Did you know that while you're in the mini-assembler (!) mode, that ($) lets you use all of the (*) monitor features?" That information (which is quite useful if you have an Apple) can be found buried in the documentation, but it should be clearly brought out in a reference manual.

But there isn't a reference manual. There is an updated version of the old (blue) manual, and a manual for Applesoft that parallels its style and format. These manuals are well-written textbooks for beginners; they are not reference manuals. For instance, while working out the bugs of a new program, one might want to look up the use and effects of the DIM (Dimension) command. The manual's index gives:

DIM 88-90, 94-97, 116, 117

Here's another example:

Pound sign 57, 61, 100, 106-107, 117, 120, 125

(This symbol, which is quite useful, exists in integer BASIC, but not in the floating point version.)

It has been widely noted that the designer of the keyboard contributed an outstanding goof to the Apple by mounting the RESET button next to RETURN, thus making it quite easy to hit RESET inadvertently. There have been suggestions (such as in the February 1979 Dr. Dobb's Journal) on how to remount the RESET button in a more remote position, which must leave your machine looking like hell.

Despite these deficiencies, most of which are easily corrected, the Apple still rates high and is surely a powerful and top-quality machine.

We have taken the Apple people to task in this article for their sins, which are largely of omission. In all probability, the list of goofs, weaknesses, and other deficiencies for the competitive machines would be longer. We should add the comments of a satisfied user:

> "Apple tries. Although some of their software and hardware goes out the door before it's fully "road tested," the users are eventually given the correct solutions to any problems. The Apple II BASIC Programming Manual is a neat, colorful, and useful text for the BASIC novice. If this document hints of products to come, then Hooray Apple! Apple could have chosen a keyboard requiring pin-point precision typing, but instead chose a "normal-sized" keyboard with a light touch conducive to the many hours of late-night work the average Apple user endures."

Radio Shack's TRS-80

by Larry Clark

Since Radio Shack introduced the TRS-80 last Spring, they have sold over 100,000 units. Although much of this success can be attributed to aggressive advertising and to control of 7000 captive retail outlets, there are also solid technical reasons behind the TRS-80's popularity.

I acquired my TRS-80 about seven months ago. Since then, I have expanded the configuration until it is now fairly complete. Although I am pleased with the system as a whole, it is not without its disappointments. This report describes both the strengths and the deficiencies of the system.

I should indicate that I have over 16 years of experience in software design, implementation, and management; my special interests are in interactive systems and user-interface design.

Hardware

Undoubtedly, many customers were first attracted to the TRS-80 by its low initial price. For $599, the system comes complete with a keyboard (which houses the Z-80 processor and memory), a 12" black-and-white display, a cassette player/recorder, a small outboard power supply, assorted cables, and an excellent manual. All of the components are packaged in silver-and-black plastic cases, which look attractive and seem sturdy enough for normal use.

The manual deserves special mention. Assuming no previous computer knowledge, it leads the reader into the system gradually, blending tutorial sections with numerous try-it-and-see examples. Marginal notes give details for more experienced users. I feel that it is one of the finest introductions to computing that I've seen.

The basic configuration includes 4K bytes of dynamic RAM and the "Level I" BASIC interpreter, which is stored on a 2K ROM. A 12K "Level II" ROM is available for an additional $99. The 16K memory option, which now includes a numeric keypad, costs $299.

The keyboard conforms closely to the ASCII layout, and has a reasonably good feel. It should not offend experienced typists, although some may miss the audible feedback. Level I software does not support keyboard rollover, so it is possible to miss characters in rapid bursts (Level II corrects this).

The display is divided into 16 lines of 64 character positions. Each character position is further divided into six smaller rectangles, making a 128 x 48 grid. The user may display text in any character position, or he may turn on, turn off, or interrogate the state of any of the smaller rectangles. Much to the dismay of many users, there is no provision for lower case, even though it is really "in there." (A one-chip modification is widely available from independent sources to enable the display of lower case, but this does not work well with Radio Shack's software.)

The cassette recorder is designed for audio, not computer, usage. As a result, its operation is somewhat slow (250 baud in Level I, 500 baud in Level II) and cumbersome. The user must manually switch from RECORD (for writing) to PLAY (for reading). In Level I, he must also position the tape rather precisely. This requires pulling the control jack out of the recorder, then using the REWIND and FAST FORWARD controls as necessary, and finally, reinserting the control jack. The Level I system has no command to verify that a CSAVE command worked. Radio Shack advises users to dump important files more than once--and hope that one of them is good! The only way to be sure a file is good is to reload it, which clobbers the system's contents. (Level II has a verification command.)

Level I BASIC

The Level I BASIC interpreter is so limited that I don't recommend it, even for people who only want it to play simple games. I suspect that it was developed strictly to keep the advertised price down and to bring the TRS-80 to market a few months sooner. Now that Level II is here, I feel it is shameful to continue selling Level I systems.

Although most BASICs are limited in variable names, Level I is worse than average. Only 26 identifiers (A through Z) are available for normal (floating point) variables. One array, A(i) is provided; this array requires no DIM statement, and is "sparse"--that is, unused elements take up no storage space.

The string facilities are even worse. Only two strings, A$ and B$, are provided. Each can be assigned up to 16 characters, via either LET or INPUT statements. However, the only allowable operation on strings is to PRINT them. They cannot be indexed, concatenated, or even compared for equality against one another or against constants.

The floating point variables used in Level I appear to carry 24 significant bits internally. They also seem to carry decimal exponents, in the range [-38,38]. Although more digits are accepted on input, only 6 significant digits are printed on output. Thus, two numbers that differ internally are printed identically.

Only the most elementary arithmetic operations are provided. Exponentiation, square root, logarithms, and trigonometric functions must be computed via subroutines.

Since the system supports only a single, sequential device (the cassette), programs that wish to update files are limited to what can be contained in RAM. It is not possible to read a record, update it, and rewrite the result, then access the next record, etc.

Level II BASIC

Level II BASIC represents the "real" TRS-80. Written for Radio Shack by Microsoft, it seems to provide a good range of features, coupled with excellent reliability. Its major enhancements include:

- an unlimited number of one- or two-character names for arithmetic variables, strings, and arrays (longer names may be used, but only the first two characters are significant);
- string lengths up to 255 characters, with provision for comparison, substringing, concatenation, etc.
- integer and double-precision floating-point variables, in addition to single-precision floating point;
- a complete set of functions, including one user-definable function;
- a simple command to perform intra-statement editing;
- the addition of ELSE constructs to IF...THEN statements;
- named cassette files, faster (500 baud) cassette transfer rate, and a command to verify the contents of a cassette file;
- keyboard rollover, to keep up with fast typists;
- automatic prompting for line numbers during program input;
- faster graphics;
- commands to trace program execution; and
- error trapping, to help "seal off" programs from user errors.

The Level II manual is clearly written, but it assumes familiarity with Level I. This makes it hard for new users to leap directly to Level II, because some of the Level I syntax is changed and because abbreviations are not accepted. It is difficult to use the manual for reference, since it lacks an index.

Peripherals

An impressive array of peripheral devices was announced almost immediately after the TRS-80 was introduced. This convinced me that Radio Shack had done some serious system design before going to market, and was a major factor in my decision to buy the system.

Most of the peripherals require use of an Expansion interface ($299), which includes a disk controller, a parallel printer interface, a second cassette interface, a real-time clock, and room to add up to 32K additional RAM ($199 per 16K). The disk controller will support up to four mini-disk drives ($499 each).

Two printers are currently supported. The Quick Printer ($499) prints at 150 lpm on 4.75" aluminum-finish paper. Although it provides some hardcopy capability, I don't consider it satisfactory for serious use--not even for program listings--since the output is small and cannot be written on. The Line Printer ($1299 with friction feed or $1559 with tractor feed) is really a Centronics 779 with a silver cover. Like other peripherals, it can be purchased without Radio Shack's nameplate and paint job for considerably less--in this case, under $1000 for the tractor feed version. The print density is variable, from 10 to 16.5 characters per inch. At maximum density, it can print 132-character lines at 21 lpm.

Using Disks

The disk drive is simply a soft-sectored Shugart SA-400 with a power supply and case. Each drive provides about 86,000 bytes of usable storage. The first disk, however, must contain the Disk Operating System (DOS) and Disk BASIC, which are provided free. This software is updated periodically; the current version is 2.1.

The presence of system software reduces the user storage on the first disk to about 55,000 bytes. During operation, these components occupy about 10K of RAM, so a 16K system becomes somewhat cramped. This also results in conflicts with assembly language software, including virtually all of the programs that Radio Shack sells (including the program editor and assembler!).

The use of DOS introduces the concept of modes. At any point in time, the system may be operating in DOS, in BASIC, or in some other program. Exiting from BASIC to DOS destroys any programs and data that were being manipulated under BASIC. This can be extremely frustrating in light of the system's tendency to return to DOS whenever a disk error is detected. (Lost Data flags seem to arise from a wide variety of causes, and always at inopportune times.) Thus, an attempt to save a program can actually cause it to be lost.

DOS includes commands to format disks, copy disks, list directories, set date and time, copy files, print files, and delete files. The file system incorporates a rather sophisticated, five-level protection scheme. DOS also includes a primitive, machine-language debugger, which only partially works as specified.

Disk BASIC provides commands to enable and disable clock interrupts. Unfortunately, the interrupts must be disabled during cassette operations, and they must be enabled during certain (unspecified) disk operations. Failure to comply can result in anything from bad reads to destruction of the diskette's data.

Evaluation

On the whole, I feel that the TRS-80 deserves the success it enjoys. For under $2000, a user can purchase a Level II system with the Expansion Interface, 32K of RAM, and a single disk. Except for the lack of lower-case, the hardware seems well designed and carefully matched. The recent announcements of new peripherals suggest that even better things are forthcoming.

The system software is less consistent than the hardware. Although Level I BASIC is something of a joke, Level II is a sheer joy. At the present time, the sophisticated features of DOS tend to be eclipsed by abominable error-handling procedures. I remain hopeful that this situation will improve as DOS matures.

A limited amount of applications software is available from Radio Shack. This includes the renumbering program, the assembler and editor, an extensive financial package, a tutorial for Level II, and assorted games and demonstrations, all at rather reasonable prices. Some of the programs I've seen are rather good; others are downright amateurish. Unfortunately, there is no way to judge a program (or even to find out exactly what it does) except by buying it.

Other than the manuals for Level I and Level II, most of the documentation is poor, and often hard to get. DOS users are still operating from a preliminary manual for version 2.0 and an update for version 2.1. Both the real DOS manual and a reference manual that explains the hardware are supposed to be available, but I haven't found a store that has either one.

I am frustrated by Radio Shack's unwillingness to distribute listings of the system software. Many users, if they had access to the listings, might fix bugs and develop useful extensions. Since users seem anxious to publish what they've accomplished, Radio Shack is missing out on a wealth of free software help.

Although the TRS-80 is far from perfect, I haven't found anything I like better at near the price. With so many units in the field, and a large organization supporting the effort, I am confident that the system will continue to improve.

[Mr. Clark's years of experience include a period when he was the man in charge of the JOSS system at the RAND Corporation. He was featured in the 1965 film "JOSS," along with JOSS's inventor, J. Clifford Shaw. We believe that this is the first published review of the TRS-80 written by someone who is competent to compare it to other systems.]

HALLmarks

In our issue number 72, Problem 256 was presented in the article "Solution Evolution--2." The problem was:

> Three variables, A, B, and C are to be progressively incremented with the mantissas of successive square roots. The initial values of the three variables are thus:
>
> A = .23606797749 (from the square root of 5)
> B = .41421356237 (from the square root of 2)
> C = .73205080756 (from the square root of 3)
>
> Each new mantissa (the next one is .44948974278) is to be added to the variable whose contents at that time is the smallest (thus, the mantissa of the square root of 6 is to be added to A).
>
> What we want at each stage is the variance of the contents of A, B, and C, and specifically we want to record successively lower values of the variances.

A possible method of solution was given, together with the first nine occurrences of new lower variances.

Bob Hall, Arleta, California, writes:

"In my opinion, the most significant shortcoming of programmable calculators is their slow execution speed, and this problem is a good example of it. Time after time I have run into such cases, and no amount of programming finesse can help when a factor of 100 or more in speed is what is really needed.

"With calculations only through N = 141,000, I would like to state a few conjectures about this problem, the truth (or falsity) of which may better be determined by those persons with access to faster machines.

"The attainment of close agreement between the three numbers is really more likely with smaller differences between successive square roots, rather than with smaller fractional values for numbers slightly greater than perfect squares.

> (1) Therefore, new sets with reduced variance may occur with large as well as small fractional parts of the square root of N.

"Since the differences are continuously decreasing, successive fractional parts become ever more nearly equal. Since adding nearly equal numbers to nearly equal numbers yields nearly equal numbers, it follows that:

(2) New sets with reduced variance are likely to occur in "triples" (i.e., N, N+3, N+6, etc.),

But the successive values of new lower variances appear to decrease faster than the successive differences.

(3) Therefore, new sets with reduced variance will occur less and less frequently (i.e., at ever-increasingly greater differences between values of N at which they occur).

(4) Thus it appears that such new sets in sequences (such as "triples") may never occur at all, since the problem always seeks lower variances within each such sequence.

"Analysis of the outputs shows that the fractional parts are small and decreasing, but not minimal. This seems to disprove (1). I do not understand why. At the moment, I have insufficient data to verify (or refute) (2) and (4). (3) appears to be true.

"The accumulated calculating time from N = 5 through N = 141,000 on my HP-29C was about 170 hours. This is an interesting problem, but my solution was not too satisfying due to the slow speed of execution.

(At this point, Mr. Hall had accumulated the following results)

N	Variance	N	Variance
6	.017680978	7587	.0014302934
18	.071521011	9044	.0013824780
29	.014508327	9428	.0010130340
54	.012894753	12120	.0009927900
86	.007127451	21927	.0009161700
634	.006227937	22524	.0007190370
686	.003234124	23740	.0006783690
738	.003212868	33516	.0006501906
1949	.001910437	43709	.0005844504
3150	.001526976	44128	.0004510290
		67113	.0003671490
		91843	.0002547379
		134727	.0002264970
		139914	.0002121399

A later communication from Mr. Hall goes on:

"My improved algorithm works very well and summarizes thus:

> The midsum value is not stored inasmuch as it is always reset to zero after each summation by appropriate adjustment of minsum and maxsum. This saves on sorting operations while simultaneously keeping magnitudes of minsum and maxsum low (ie, less than one), simplifies calculation of the variance, and reduces program statements required.

For example, the given initial values are:

$$\text{MIN} = \text{mantissa of } \sqrt{5} = 0.236...$$

$$\text{MID} = \text{mantissa of } \sqrt{2} = 0.414...$$

$$\text{MAX} = \text{mantissa of } \sqrt{3} = 0.732...$$

The variance is the same if:

$$\text{MIN} - \text{MID} \longrightarrow \text{MIN} \quad (= 0.236 - 0.414 \cong -0.178)$$

$$\text{MID} - \text{MID} \longrightarrow \text{MID} \quad (= 0.414 - 0.414 \cong 0)$$

$$\text{MAX} - \text{MID} \longrightarrow \text{MAX} \quad (= 0.732 - 0.414 \cong 0.318)$$

and these are my new starting values.

"In the running program, $\text{FRAC}(\sqrt{N}) + \text{MIN} \longrightarrow \text{MIN}$. Thus, the "MIN" may, in reality, be greater than zero, and, if greater than zero, it may also be greater than MAX. The following logic corrects this:

$$\text{MIN} + \text{FRAC}(\sqrt{N}) \longrightarrow \text{MIN}$$

IF MIN greater than zero, then

If MIN > MAX, then

$$\text{MAX} \longrightarrow \text{TEMP};\ \text{MIN} \longrightarrow \text{MAX};\ \text{TEMP} \longrightarrow \text{MIN}$$

$$\text{MAX} - \text{MIN} \longrightarrow \text{MAX};\ 0 - \text{MIN} \longrightarrow \text{MIN}$$

(Exchange MAX and MIN; "Adjust" to make MID = 0)

This "adjustment" of MIN and MAX is derived by "resetting" MID to 0, viz...

If MIN 0: means that MIN and MID have been exchanged by the summation.

If MIN MAX: means that MID and MAX have also been exchanged.

The algorithm given above, in reality, reverses these to make MIN < MID (=0) < MAX.

"I programmed this algorithm on my HP-29C calculator and obtained the following results:

N	Variance
142167	.00003345046
157648	.00002849966
255070	.00002617399
351697	.00002362866

"Then, while attending the Sperry-Univac programming course in Minneapolis, I had the opportunity to accomplish the same logic in a PASCAL program which was then executed on a Univac II...reaching, in one run, N = 10,000,000. The largest result had N = 6,345,461 with a variance of .0000040472. Several of the results achieved earlier on the hand calculator were skipped by the PASCAL program; I do not know why."

Mr. Hall notes that the ratio of computer speed to calculator speed is of the order of 16000 to 1. With regard to his conjectures, he concludes that (1) and (2) are false and (3) and (4) are true. Finally, he suggests that the boxed quotation we ran at the beginning of the original article be changed to be:

After a program is written, debugged, thoroughly tested, and run for a while--only then do we know just how it should have been written.
--Bob Hall's version

...or Not To Recurse

Picture an endless stream of numbers:

A B C D E F G H I ...

For each set of three consecutive numbers, we form the new numbers:

$$\frac{B+C}{A} \qquad \frac{E+F}{D} \qquad \frac{H+I}{G} \quad \ldots\ldots$$

in other words, for each set of three numbers, divide the first of them into the sum of the other two.

This gives a new endless series of numbers, and the same process is to be applied to that set to produce a new set, to which the same process is applied, and so on.

If the first set is the natural numbers, then the whole process is rather dull and predictable--every term at every level tends toward 2.00 rather quickly. We can patch that deficiency later, but for now let's consider how the various sets could be calculated.

At each level, when three numbers have accumulated, then one number at the next lower level can be calculated. Thus, for any given level, a subroutine could be written to do the following...

Set up four buckets, P, Q, R, and C, all initially set to zero. When this subroutine is called, there will be a result available from the next higher subroutine. This result is to be put into P, Q, or R, whichever is currently zero. Bucket C is a counter, and it is now incremented by one.

If C is less than 3, control returns to the main program. If C = 3, then the basic calculation is performed:

$$XX = \frac{Q+R}{P} \ ,$$

P, Q, R, and C are reset to zero, and the subroutine at the next lower level is called to receive the new result.

The number of such subroutines to be written depends on the depth to which the calculation is desired. If the first set of calculated values is called level zero, each new entry at that level will be obtained when three elements of the original generating set of numbers is used. Succeeding levels, then, will add an element as follows:

level	1	for each	9	of the original set
	2		27	
	3		81	
	4		243	
	5		729	
	6		2187	
	7		6561	and so on...

Thus, to find the first dozen elements at level 10, calculations at level zero will have to be made through 2,125,764 elements of the generating set.

To go to level 10, then, would call for writing eleven subroutines--and they would be identical in their form; that is, each of them would have precisely the same instructions. The only difference between them would be that each of them would have its own set of buckets P, Q, R, and C.

In other words, the situation seems to call for one subroutine, called recursively.

At this point, perhaps we should refer again to the diatribe of Forman Acton (in Numerical Methods That work):

> "...it is probably time to commit a public heresy by denouncing recursive calculations. I have never seen a numerical problem arising from the physical world that was best calculated by a recursive subroutine--that is, by a subroutine that called itself. I admit the idea is cute and, once mastered, it tends to impel its owner to apply it wherever possible--all questions of appropriateness aside."

Consider also the article "Recursive Programming in BASIC" by Herbert Dershem (in the April 1979 issue of Personal Computing), in which a 15-line program is given that will demonstrate, on any system that supports BASIC, the extent to which one can use recursive subroutining.

..... The first dozen terms at each level .....

Level Number												
0	6.0000	3.0000	2.5714	2.5000	2.3077	2.2500	2.2631	2.1818	2.1600	2.2143	2.1290	2.1176
1	1.0952	2.2231	2.3603	2.3694	2.4322	2.4463	2.4345	2.4622	2.4671	2.4420	2.4739	2.4764
2	6.0109	2.9030	2.8464	2.8461	2.8215	2.8169	2.8224	2.8119	2.8103	2.8205	2.8082	2.8074
3	1.4556	3.0352	3.0549	3.0546	3.0636	3.0653	3.0632	3.0671	3.0677	3.0639	3.0684	3.0687
4	6.9320	3.3159	3.3085	3.3086	3.3053	3.3046	3.3054	3.3039	3.3037	3.3051	3.3034	3.3033
5	1.6769	3.5090	3.5118	3.5118	3.5130	3.5132	3.5130	3.5135	3.5136	3.5136	3.5137	3.5137
6	7.7648	3.7093	3.7083	3.7082	3.7078	3.7077	3.7078	3.7076	3.7076	3.7078	3.7076	3.7076
7	1.8568	3.8874	3.8878	3.8878	3.8880	3.8880	3.8880	3.8881	3.8880	3.8881		

```
100 DIM A(50)
110 FØR I = 1 TØ 50
120 A(I) = 0
130 NEXT I
140 DIM Y(3)
150 FØR I = 1 TØ 3
160 Y(I) = I
170 NEXT I
180 YY = 1

300 XX = (Y(2) + Y(3) + YY)/Y(1)
305 PRINT , "MAIN";"bb"; XX
310 YY = 1
320 GØSUB 1000
330 FØR I = 1 TØ 3
340 Y(I) = Y(I) + 3
350 NEXT I

400 A(T) = A(T) + 1
405 IF A(Q) = 0 THEN 445
410 IF A(R) = 0 THEN 435
415 IF A(S) = 0 THEN 425
420 GØTØ 500
425 A(S) = XX
430 GØTØ 450
435 A(R) = XX
440 GØTØ 450
445 A(Q) = XX
450 IF A(T) < 3 THEN 300

500 XX = (A(R) + A(S) + YY)/A(Q)
510 PRINT YY;"bb"; XX, Y(1)
520 A(T) = 0
530 A(Q) = 0
540 A(R) = 0
550 A(S) = 0
560 A(U) = 0
570 YY = YY + 1
575 IF YY > 9 THEN 300
580 GØSUB 1000
590 GØTØ 400

1000 Q = 5*YY - 4
1010 R = 5*YY - 3
1020 S = 5*YY - 2
1030 T = 5*YY - 1
1040 U = 5*YY
1050 RETURN

2000 END
```

A program written in Applesoft BASIC to implement the problem described in the text.

This problem was eventually coded in BASIC without using subroutines at all. The code is given here (the subroutine at 1000 is used simply to avoid repetitive address modification). The accompanying chart shows the first twelve elements for the first 8 levels. To make the calculation less dull (statements 300 and 500), the current level number is added to the numerator of the fractions. Within the limits of the computer run represented by the chart, the even numbered levels decrease and the odd numbered levels increase, which is interesting and puzzling.

The objective here was to experiment with recursion, even though the end result did not need it.

The final process is not unduly complex and is well defined, at least via the BASIC program. It would be difficult to produce the results shown without a computer and it would probably be impossible to produce any results at all for level 25 without extensive use of a fast machine.

Challenge Problem No. 6

The following numbers:

88 90 139 204 542 691 696 772 954 and 960

have something in common. What is it?

Similarly, the following list of numbers have a common element:

91 220 254 329 336 370 468 499 607 747 902.